COBB

INSIDE DOS

Tips & techniques for MS-DOS & PC-DOS

March 1993
Vol. 4 No. 3

A pair of batch files can protect your computer from the casual snoop

By Van Wolverton

If you use your computer at home, you probably don't worry much about unwelcome snoops browsing through your files or programs. But if you use a computer at work or in a dorm room, chances are that once in a while you wish for some way of keeping others from using your system or poking through your files.

If you have a big budget, you might install a device that allows you—or anyone else—to use your computer only after entering the correct password. Or, you might choose a system that requires a key or magnetic card to start your PC. Even if you can afford these devices, your company may not allow you to add security hardware to its systems.

You can, however, guard your system against the casual snoop with nothing more than DOS itself. The two batch files described this month blank the screen and ask for the correct entry of a password to resume normal operation. They're based on a similar technique devised by an old friend, Dave Hill, of Woodinville, Washington.

The security batch files also show ways of saving and restoring your command path or prompt and using ANSI.SYS commands to control display attributes and clear the screen. You might find these techniques useful in other batch files as well. In this article, we'll first look at saving and restoring the prompt; then, we'll examine the ANSI.SYS commands for blanking the DOS screen. Finally, we'll show how you can use these commands in the pair of security batch files.

Saving and restoring the prompt

Environment variables give you a convenient way of saving and restoring your system prompt. The technique is straightforward: When DOS runs a batch file and finds the name of an environment variable enclosed in percent signs, it replaces the variable's name with the variable's value.

For example, suppose your standard system prompt is *pg*. (This prompt displays the current drive and directory, followed by the greater than sign.) When you run a batch file, DOS will replace any references to the environment variable *%prompt%* with *pg*. That's not very exciting, but suppose DOS found this command in a batch file:

```
set promptrs=%prompt%
```

(The command will work only in a batch file because DOS won't replace environment variables typed at the command line.) DOS would first replace *%prompt%* with the current prompt definition and then carry out the SET command. In other words, DOS would carry out the instruction as

```
set promptrs=$p$g
```

After the batch file completed, the DOS environment would contain an environment variable named PROMPTRS (the name could mean *prompt restore*). This variable's value is the current prompt definition.

IN THIS ISSUE

Suppose your prompt definition was *pg* and you carried out the SET command just described. When you displayed the contents of the environment by typing *set*, you'd see the following lines included in the listing:

```
PROMPT=$p$g
PROMPTRS=$p$g
```

Once you've saved the current prompt in the variable PROMPTRS, you can save the prompt and restore it with the following batch file command:

```
set prompt=%promptrs%
```

If you later want to get rid of the variable PROMPTRS to save environment space, you can use the command

```
set promptrs=
```

Now that we've looked at ways of saving and restoring the path using environment variables, let's look at the ANSI.SYS commands that can blank and restore the screen.

Controlling the screen and keyboard with ANSI.SYS

ANSI.SYS is a device driver that controls the screen and keyboard. (The article "Trading Key Assignments with ANSI.SYS," on page 6, reviews the steps for installing the ANSI.SYS device driver.) Like all ANSI.SYS commands, the commands you'll use to blank the screen begin with the Escape character—ASCII 27. You can't type the Escape character because whenever you press the Escape ([Esc]) key, DOS erases the command line and starts over. The PROMPT command, however, lets you represent the Escape character with other characters that you *can* type—*$e*—so you can use the PROMPT command to enter ANSI.SYS commands.

All ANSI.SYS commands start with the Escape character followed by a left bracket, and they end with a letter (the letter identifies the specific command). The body of the command consists of one or more numbers separated by semicolons. You'll use the three ANSI.SYS commands shown in Table A in the security batch files (in the table, *$e* represents the Escape character, just as it does in the PROMPT command).

Table A

Code	Effect
$e[0;8m	Makes text invisible
$e[2J	Clears the screen
$e[0m	Makes text visible

Combined with the PROMPT command, these ANSI commands will allow you to blank and restore your screen.

INSIDE DOS™

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group.

Prices
Domestic ..... $49/yr. ($6.00 each)
Outside U.S. ..... $69/yr. ($8.50 each)

Phone
Toll free ..... (800) 223-8720
Local ..... (502) 491-1900
Customer Relations Fax ..... (502) 491-8050
Editorial Department Fax ..... (502) 491-4200

Address
You may address tips, special requests, and other correspondence to:

The Editor, *Inside DOS*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to:

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Postmaster
Second class postage is paid in Louisville, KY. Send address changes to:

Inside DOS
P.O. Box 35160
Louisville, KY 40232

Back Issues
To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $6.00 each, $8.50 outside the U.S., and you can pay with MasterCard, VISA, Discover, or American Express, or we can bill you. Please identify the issue you want by the month and year it was published. Customer Relations can also provide you with an issue-by-issue listing of all the articles that have appeared in *Inside DOS*.

Staff
Editor-in-Chief ..... Suzanne Thornberry
Contributing Editors ..... Van Wolverton
David Reid
Editing ..... Martha A. Clayton
Elizabeth Welch
Production Artist ..... Julie Jefferson
Design ..... Karl Feige
Publications Manager ..... Tara Dickerson
Circulation Manager ..... Brent Shean
Publications Director ..... Linda Baughman
Editorial Director ..... Jeff Yocom
Publisher ..... Douglas Cobb

Advisory Board
Earl Berry Jr.
Tina Covington
Marvin D. Livingood

The commands that make the text invisible and visible end with lowercase *m*; the *m* identifies the ANSI.SYS command that controls monitor attributes, such as color or blinking. The command that clears the screen is named *J*.

The security batch files

The method of protecting your system against the casual snoop is simple: One batch file, called SLEEP.BAT, makes the text onscreen invisible. The second batch file—we'll call it PASSWORD.BAT in this article—carries out the ANSI.SYS command that makes text visible. You should name your version of PASSWORD.BAT with any string containing up to eight characters (which may include letters, numbers, and even some symbols, such as _, &, #, !) followed by the BAT extension. The name you choose instead of PASSWORD.BAT will be your password. For example, you could name your version RAW#HIDE.BAT.

You can create both batch files with the DOS Editor or any word processor that allows you to save plain, ASCII-only text. Be sure to keep the two batch files in a directory that's on your command path. (We usually use the C:\BATCH directory.)

SLEEP.BAT, shown in Figure A, does most of its work with only two commands, SET and PROMPT.

Figure A

```
@echo off
rem This is SLEEP.BAT
set promptrs=%prompt%
prompt $e[0;8m$e[2J
```

SLEEP.BAT makes the information on your screen invisible.

As you can see, SLEEP.BAT begins with the standard *@echo off* command, followed by a remark describing the batch file. Then, the SET command saves your current prompt definition in the environment variable PROMPTRS. The last line uses the PROMPT command to introduce two ANSI commands. The first, *$e[0;8m*, makes text invisible; the second, *$e[2J*, clears the screen. After SLEEP.BAT carries out these commands, the screen blanks.

As you can see in Figure B, the second batch file isn't much longer.

Figure B

```
@rem This filename is your password.
prompt $e[0m
mode co80
@echo off
set prompt=%promptrs%
set promptrs=
```

PASSWORD.BAT restores your screen. You should choose your own code word instead of the name PASSWORD.

PASSWORD.BAT begins with a REM command, which lets you insert a note about the batch file. The @ symbol that precedes the remark prevents DOS from displaying the line onscreen. (We didn't use the *@echo off* instruction, because we need echo on for the PROMPT and MODE commands.) Next, the PROMPT command carries out the ANSI.SYS command that restores the text. The MODE command corrects any anomalies in screen or cursor behavior arising from previous ANSI commands. Next, the first SET command restores the prompt definition that SLEEP.BAT stored in the temporary environment variable PROMPTRS. Since PROMPTRS has now served its purpose, the second SET command deletes the environment variable. (If you want your system to change to a particular directory or start a program when you type the password, put those commands at the end of the PASSWORD.BAT batch file.)

Once you've created both batch files and named the second batch file whatever you'd like your password to be, you're ready to set your system to boot into the sleep mode. To do so, simply edit your AUTOEXEC.BAT file and add SLEEP as the last command. This step will prevent someone from gaining control of your system by pressing [Ctrl][Alt][Del] to restart it. Rebooting the system will return it to the state it was in before you pressed [Ctrl][Alt][Del].

Of course, you don't have to reboot your system to take advantage of the security batch files. Whenever you're going to leave your system for a while, just type *sleep*. Not only are you giving your system some security against unwanted users, but you're also protecting the screen against burn-in. When you want to use your system again, just type the password.

A note for restoring color

The PASSWORD.BAT file shown in Figure B makes text visible, but it doesn't restore any colors you may have selected for your screen. For example, you might use a COLOR.BAT file like the one we described in the article "Using the PROMPT Command to Color Your DOS Screen" in the May 1992 issue of *Inside DOS*. In this case, you can probably use the COLOR.BAT file instead of PASSWORD.BAT. To test COLOR.BAT, first run it, then type *color* and press [Enter]. If COLOR.BAT works for restoring your screen, you can rename COLOR.BAT to the password you'd like to use. Or, if you've created other batch files that call COLOR.BAT, you can copy COLOR.BAT as the filename you want to use for a password. For example, you could enter the command *copy c:\batch\color.bat c:\batch\password.bat*.

If running COLOR.BAT doesn't restore your screen, you can copy the PROMPT command that sets the screen colors from COLOR.BAT and add a CLS command to the end of PASSWORD.BAT. You can replace the command *prompt $e[0* with the command that sets the colors.

For example, if your COLOR.BAT file uses the command *prompt $e[1;33;44m* to select a high-intensity yellow foreground on a blue background, your PASSWORD.BAT file would look like the one in Figure C.

Figure C

```
@rem This filename is your password
prompt $e[1;33;44m
mode co80
@echo off
set prompt=%promptrs%
set promptrs=
cls
```

You can set the codes that set colors in PASSWORD.BAT.

Be environmentally aware

Because SLEEP.BAT saves the current prompt definition in a new environment variable before it deletes the prompt, the DOS environment must be large enough to temporarily hold two copies of the prompt definition. In most cases, the environment size isn't a worry when your prompt definition is short (such as *pg*). However, environment space could be a problem if you use a long prompt definition.

If you see the *Out of environment space* message or if the system behaves strangely after you type the SLEEP command, edit the CONFIG.SYS file and increase the environment size specified with the /E parameter of the SHELL command. If there is no SHELL command in

Special space character can help keep files private

Here's a quick tip for further frustrating the casual snoop: You can include a special space character in the names of important files. This special character is known as ASCII 255. You create it by simply holding down the [Alt] key and typing *255* on the numeric keypad.

DOS will display the character as a blank space. However, you can make the space less noticeable by placing it at the end of the root name of the file. That's because the DIR command usually displays root names in one column and extensions in another. For example, suppose you have a confidential file named SALES93.XLS in your C:\DATA directory. If you issue the command *dir c:\data\sales*.*,* DOS might display the following listing:

```
SALES91      XLS      10228      03-01-93   12:28p
SALES92      XLS      86008      01-07-93    4:51p
SALES93      XLS      82760      01-19-92    9:38a
```

You add the special space character to the end of the SALES93.XLS filename by changing to the C:\DATA directory and issuing the RENAME or REN command:

```
C:\DATA>ren sales93.xls sales93[Alt]255.xls
```

(We've shown the [Alt]255 keystrokes in black because DOS will display a space in their place.)

Better still, you can rename all the files with one command by using the ? wildcard character. The ? wildcard can represent any character that occurs where you type the wildcard. Since each sales spreadsheet contains the sequence *sales9* followed by another digit, you can use the ? wildcard at the end of the filename. You can represent the group of spreadsheets as *sales9?*, then add the space character to the end of each filename by issuing the following command:

```
C:\DATA>ren sales9?.xls sales9?[Alt]255.xls
```

After issuing the command, the names of the three sales spreadsheets will have the [Alt]255 space character before the extension. Issuing a simple DIR command, however, won't reveal the space. Instead, the directory listing will look the same as in our previous example. However, you—or a snoop—could see the space in the filenames if you use the ATTRIB command or if you issue the DIR command with the /B ("bare") or /W ("wide") switches. For example, here's how DOS would display the names of the sample spreadsheets when you issued the command *dir c:\data\sales*.* /b*:

```
SALES91 .XLS
SALES92 .XLS
SALES93 .XLS
```

In this directory listing, a space precedes the XLS extension. To copy or delete the file from the DOS prompt, a snoop either would need to know how to create the [Alt]255 space character or would have to use wildcards in the COPY or DEL command.

The technique of using the unexpected space in filenames has another weakness: Snoops who start the DOS Shell, Windows, or another program with graphical file management can copy, delete, or open the file by selecting it from a list. If you want to protect files from this route of entry, you'll need to password-protect them from within the application.

CONFIG.SYS, add one that looks like this:

```
shell = c:\command.com c:\dos /e:512 /p
```

If adding this SHELL command doesn't solve the problem, increase the environment size to 640 and try again. If that still doesn't do it, increase it again to 768—that should be more than enough.

This protection isn't bulletproof

Unfortunately, this batch file method of keeping someone from using your system is just as easy to defeat as it is to implement. A knowledgeable snoop might type the ANSI.SYS command that makes the text visible. Even with the SLEEP command in your AUTOEXEC.BAT file, someone can still use your system by putting a DOS system diskette in drive A: and restarting the system.

But these methods of breaching your security, although simple to carry out, aren't well known by the average user. If you create these two batch files and put SLEEP in your AUTOEXEC.BAT file, you can leave your system unattended with some assurance that your files are safe from the casual snoop.

Contributing editor Van Wolverton is the author of the best-selling books Running MS-DOS 5 *and* Supercharging MS-DOS. *Van, who has worked for IBM and Intel, currently lives in Alberton, Montana.*

SECURITY TIP

Suspend your path for more security

In the accompanying article, Van Wolverton shows you how to create two batch files to help keep your files secure from the casual snoop. Blanking the screen will frustrate most users who try to issue a directory command to see what files are on your hard disk. However, some snoops might try to enter the name of a program that they also use. Since practically all commercial programs control the monitor directly, they will bypass the ANSI commands you used to blank the screen. The snoop could then access your files through the program.

One way of preventing a snoop from using an application is to rename the executable file—the file that runs the application. But renaming an executable file can have unwanted effects: Batch files that refer to the executable file will produce errors until you revise them with the new name. If you've set up the program through the DOS Shell, you'll need to revise its Program Item Properties dialog box to include the new filename. You'll have to carry out similar steps in Windows.

A simpler way of preventing the casual snoop from using your programs is to remove the program's directory from your path. If the snoop tries to run the program without first changing to its directory, the program command will generate the message

```
Bad command or filename
```

When you suspend the path as part of SLEEP.BAT, even this message will be hidden.

You can adapt Van's technique for saving and restoring your prompt to save and restore your path. We suggest you leave only the directory containing your PASSWORD.BAT file in the path. For example, if your PASSWORD.BAT file is stored in the C:\BATCH directory, you could add to your SLEEP.BAT file the last two lines shown in red in Figure A.

Figure A

```
@echo off
rem This is SLEEP.BAT
set promptrs=%prompt%
prompt $e[0;8m$e[2J
set pathrs=%path%
path=c:\batch
```

You can modify SLEEP.BAT to save the current path in the PATHRS variable, then set the path to a single directory.

The first new line saves the current path as the PATHRS environment variable. The second new line sets the path to the C:\BATCH directory only. (If you don't want snoops to have access to your batch files, you might set up a special directory that contains your security batch files and limit your path to that directory.)

Figure B

```
@rem This filename is your password.
prompt $e[0m
c:\dos\mode co80
@echo off
set prompt=%promptrs%
set promptrs=
set path=%pathrs%
set pathrs=
```

Adding two lines to PASSWORD.BAT restores your path and deletes the PATHRS environment variable.

Next, you'll need to modify PASSWORD.BAT to restore your normal system path. Figure B shows in red the lines you should add. The first new line sets the path to the value stored in the PATHRS variable; the second new line deletes the PATHRS variable. Note also that you need to add the path *c:\dos* to the MODE command.

Trading key assignments with ANSI.SYS

Have you ever had a frustrating exchange like this with DOS when you're trying to format a blank diskette?

```
C:\>format a;
Parameter format not correct - a;
```

Of course, DOS expects you to type a colon after the drive letter, but you might forget to hold down the [Shift] key when you try to enter the [Shift]; key combination that produces the colon. Since unforgiving DOS won't overlook your typo, you'll end up with a semicolon—and an error message.

If you frequently type one symbol when you really want another, you may be interested in a technique that lets you reassign (or *remap*) keys to other characters. As we'll show in this article, you can reassign a key by using ANSI codes along with the PROMPT command.

Note, however, that the key assignments you make are usually in effect only at the DOS prompt and in Edlin. The DOS Editor and most applications will ignore your remapping.

Installing the ANSI.SYS driver

ANSI.SYS is a special kind of file known as a *device driver*. The ANSI.SYS driver can help you control your monitor or keyboard. To take advantage of the ANSI.SYS driver, you must install it through your CONFIG.SYS file. If the ANSI.SYS file is in your C:\DOS directory, you can install the driver by placing the following line in your CONFIG.SYS file:

```
device=c:\dos\ansi.sys
```

If you've configured your system to load device drivers high, you can install ANSI.SYS with a DEVICEHIGH statement:

```
devicehigh=c:\dos\ansi.sys
```

Once you've added one of these statements to your CONFIG.SYS file, DOS will load the ANSI.SYS driver each time you boot up. You can then use ANSI codes to control your monitor or keyboard.

A special PROMPT command

Although there are other ways to send ANSI codes to DOS, the PROMPT command provides the easiest method. The command for redefining a key takes the form

```
C:\>prompt $e[defaultcode;newcodep
```

In this command, *defaultcode* is the default code assigned to the key you want to remap. You can look up these codes in Table A. (Note that Table A shows only the unshifted characters in the Key column.) *Newcode* is the code for the new character you want to assign to the key. As you can see, you introduce the ANSI codes with an escape sequence—*$e[*. Then, you type the default code, a semicolon, the code for the character you want to assign to the key, and *p* to end the command.

In many cases, you can save yourself from looking up at least one of the codes. Instead of typing the *defaultcode* or *newcode*, you can enclose in quotation marks the character you want to specify. The syntax takes the form

```
C:\>prompt $e["defaultkey";"newkey"p
```

where *defaultkey* is the character the key normally produces and *newkey* is the character you want the key to produce instead. (As we'll see later, placing the characters in quotation marks won't work when you want to reset a key you've already redefined.)

Let's look at a simple—if somewhat unrealistic—example of the two ways of redefining a key. If you wanted to define the lowercase letter *a* (code 97) as the lowercase letter *b* (code 98), you could enter the following PROMPT command:

```
C:\>prompt $e[97;98p
```

Or, you could enclose the letters in quotation marks:

```
C:\>prompt $e["a";"b"p
```

Entering either of these PROMPT commands will wipe out the system prompt you have in effect. You can restore your favorite system prompt by entering it after you've redefined the keys. For example, if you use the standard *pg* prompt, which displays the path followed by a greater than sign, you can reset your path by entering another PROMPT command:

```
C:\>prompt $p$g
```

If you place the PROMPT commands for your key assignments in the AUTOEXEC.BAT file or another batch file, you can simply place the *prompt pg* command at the end of the batch file. (Van Wolverton offers another technique for saving and restoring the prompt in his article "A Pair of Batch Files Can Protect Your Computer from the Casual Snoop," which begins on page 1.)

Now that we've shown you the basics of using the PROMPT command to redefine a key, let's look at some more realistic applications for the technique.

Table A

Key	Code	[Shift]	[Ctrl]	[Alt]
'	39	34	-	0;40
,	44	60	-	0;51
-	45	95	31	0;130
.	46	62	-	0;52
/	47	63	-	0;53
0	48	41	-	0;129
1	49	33	-	0;120
2	50	64	-	0;121
3	51	35	-	0;122
4	52	36	-	0;123
5	53	37	-	0;124
6	54	94	-	0;125
7	55	38	-	0;126
8	56	42	-	0;127
9	57	40	-	0;128
;	59	58	-	0;39
=	61	43	-	0;131
[	91	123	27	0;26
\	92	124	28	0;43
]	93	125	29	0;27
`	96	126	-	0;41
a	97	65	1	0;30
b	98	66	2	0;48
c	99	67	3	0;46
d	100	68	4	0;32
e	101	69	5	0;18
f	102	70	6	0;33
g	103	71	7	0;34
h	104	72	8	0;35
i	105	73	9	0;23
j	106	74	10	0;36
k	107	75	11	0;37
l	108	76	12	0;38
m	109	77	13	0;50
n	110	78	14	0;49
o	111	79	15	0;24
p	112	80	16	0;25
q	113	81	17	0;16
r	114	82	18	0;19
s	115	83	19	0;31
t	116	84	20	0;20
u	117	85	21	0;22
v	118	86	22	0;47
w	119	87	23	0;17
x	120	88	24	0;45
y	121	89	25	0;21
z	122	90	26	0;44

You can find the ANSI codes for keys by themselves and in combination with the [Shift], [Ctrl], or [Alt] keys.

Defining a key as an extended character

Suppose you want easier access to the nonprinting space character ([Alt]255). (We described using this character in filenames in the article "Special Space Character Can Help Keep Files Private" on page 4.) Let's also suppose you rarely use the ` (code 96) character at the top left of your keyboard.

To redefine the ` character as the nonprinting space, you begin by typing *prompt $e[*, as with any ANSI prompt command. Then, you type the ` symbol, enclosed in quotation marks, followed by a semicolon. Next, to choose the nonprinting space character, you type a quotation mark, hold down the [Alt] key, and type *255* on the numeric keypad, another quotation mark, and the letter *p*. DOS will display the [Alt]255 character as a space, as shown below:

```
C:\>prompt $e["`";" "p
```

Once you've issued this command, you can create the nonprinting space character by typing ` instead of entering [Alt]255.

Redefining the semicolon

As we mentioned, it's easy to type a semicolon instead of the shifted colon character needed for most DOS commands. If you frequently make this typo, you might switch the semicolon and colon keys. To do so, you first assign the semicolon to the colon key (or, more precisely, to the [Shift]; key combination), as shown below:

```
C:\>prompt $e[":";";"p
```

Since the ANSI code for the colon is 58 (that is, the code for the semicolon key plus the [Shift] key) and the code for the semicolon is 59, this command is the equivalent of entering *prompt $e[58;59p*.

Once you've entered this command, DOS will display a semicolon whether you type the ; key alone or with the [Shift] key. In other words, you'll have two ways of typing a semicolon but no way to type a colon. To remedy this, you can assign the unshifted ; key—the original way of creating a semicolon—to the [Shift]; keystroke. Because you've already redefined the semicolon key, however, you'll need to use ANSI codes to specify the keys:

```
C:\>prompt $e[59;58p
```

After beginning the escape sequence with *$e[*, you'll type the default code for the colon (59), a semicolon as a separator, the default code for the semicolon (58), and the letter *p*.

Now that you've entered the two prompt commands to redefine the keys, you can enter the command

```
C:\>prompt $p$g
```

to restore your normal system prompt. For the rest of your session, you can press the unshifted ; key to create a colon and [Shift]; to create a semicolon.

If you find that you enjoy this keyboard arrangement, you can add the two PROMPT commands to your AUTOEXEC.BAT file. Be sure to place them before the command that sets your normal system prompt, such as *prompt pg*.

Resetting the keys

You can reset a key combination you've redefined by entering another PROMPT command. This time, however, you'll need to specify the original ANSI code twice. For example, to tell DOS to return the ; key to its original function, you'd specify the code 59 twice, separated by a semicolon:

```
C:\>prompt $e[59;59p
```

Looking up and typing the ANSI codes can become tedious when you have many keys to reset. An easier method is to remove the PROMPT commands you no longer need from your AUTOEXEC.BAT file. If you might need them again, just type *rem* and a space in front of the commands you want to exclude. When you reboot, DOS won't carry out the instructions preceeded by the REM command. Of course, you can redefine the keys yet again by removing the REM and the space in front of the PROMPT commands, saving the AUTOEXEC.BAT file, and rebooting.

Notes

The key codes and techniques we've presented work with IBM-compatible keyboards. If your keyboard isn't fully compatible, you might not be able to use some of the key combinations shown in Table A, especially those using the [Alt] key.

In addition, reassigning letter-key combinations that use the [Ctrl] key may prevent you from entering some special ASCII characters. For example, you might use [Ctrl]L to echo a form feed character to the printer, as we explain in "Quick Ways to Eject a Page from Your Printer" on page 9. However, if you reassign the [Ctrl]L key combination, you can't issue the command *echo^L>prn* to eject a page from your printer. In fact, we found that redefining [Ctrl]L also prevents you from using the [Alt]12 keystroke as an alternative way of creating the form feed character. Fortunately, batch files created to echo the ^L form feed character to the printer will still work correctly.

Conclusion

In this article, we've shown how ANSI.SYS allows you to remap the keys on your keyboard. In a future article, we'll show how a similar technique allows you to redefine your function keys to carry out the tasks you need most often.

The following articles show you other techniques that take advantage of ANSI.SYS:

"Selecting Larger Display Type with ANSI.SYS Codes," August 1992

"Adding a Header Line to Your DOS Screen," June 1992

"Using the PROMPT Command to Color Your DOS Screen," May 1992

A quicker way to navigate between subdirectories

DOS uses the .. symbol to represent the parent of the current directory. For example, if you're in the C:\1993\MARCH directory, you can change to the parent directory—C:\1993—by issuing the following command:

```
C:\1993\MARCH>cd..
```

But suppose you want to change to the C:\1993\FEB directory instead of the C:\1993 parent directory. If you're in the C:\1993\MARCH subdirectory, you could switch to C:\1993\FEB by entering two commands:

```
C:\1993\MARCH>cd..
C:\1993>cd feb
```

However, you can accomplish this switch with a single command by using the .. symbol, along with the FEB subdirectory name. The previous commands are equivalent to the following command:

```
C:\1993\MARCH>cd ..\feb
C:\1993\FEB>
```

As you can see, when you enter *cd ..\feb*, DOS will make the C:\1993\FEB directory current.

Quick ways to eject a page from your printer

If you use a laser printer, you've probably found it slightly inconvenient to print a screen from DOS. When you press [Shift][PrintScreen], DOS scans the current screen and sends it to the printer. But with most laser printers (and some other types), you'll have to manually eject the page by pressing the form feed button on the printer.

Depending on the printer you use, you may have to perform up to three keystrokes to eject the page. For example, on a Hewlett-Packard LaserJet IIP, you first press the Online button to take the printer offline, then the Form Feed button, and then the Online button again to put the printer back online.

Fortunately, you can send the form feed command to the printer right from the command line. The trick is to echo the form feed character to the printer. You can create this extended ASCII character by holding down the [Ctrl] key and typing the letter *l*. DOS will display the characters as ^L:

```
C:\>echo ^L > prn
```

Although this method of ejecting a page can save you time—especially if your printer is far from your workstation—it still requires 12 keystrokes. As you might expect, you can reduce the keystrokes needed for ejecting a page by saving the command as a batch file or DOSKEY macro. Let's take a brief look at both methods.

Placing a form feed in a batch file

You can place the ECHO command in a small batch file by itself. For example, you could create a batch file named FF.BAT containing the following line:

```
@echo ^L > prn
```

As you can see, the batch file command looks much like the FORM FEED command you entered from the DOS prompt. However, you can precede the batch file command with the @ symbol to prevent DOS from displaying the command during execution.

You'll need to use a slightly different technique to create the form feed character in the batch file. When you want to create a special character in the DOS Editor, you must first press [Ctrl]P. This keystroke combination tells the DOS Editor that the next thing you type will be a code for an extended ASCII character. After pressing [Ctrl]P, you can create the form feed character by holding down the [Alt] key while you type *12* on the numeric keypad. (You have to use the [Alt]-*number* method in the DOS 5 Editor because the Editor assigns the [Ctrl]L keystroke to the Repeat Last Find command.)

You can also use the *echo ^L > prn* command in any other batch file for ejecting a page from the printer. If you've already turned off ECHO in the batch file, you don't need to include the @ symbol in the command. For example, you might write a batch file that types a file to the printer. You can add *echo ^L > prn* after the TYPE commands in order to eject the page.

As a matter of fact, using the form feed character within a larger batch file really is more efficient than creating a one-line FF.BAT file. Any batch file, no matter how small, takes up a full allocation unit (usually 2,048 bytes) on your hard disk. As we'll see in the next section, you can set up a DOSKEY macro instead of a batch file to put the form feed at your fingertips.

A form-feeding macro

The form feed technique we've shown you works as efficiently as a DOSKEY macro. Alone, the command is too short to warrant a batch file, yet it's cumbersome to type at the command line.

You can set up DOSKEY macros directly from the command line; however, you'd have to set up the macro again any time you reboot your system. That's why we suggest you place the command for creating the form feed macro in your AUTOEXEC.BAT file. Or, better still, place the command in a MACROS.BAT file like the one we described in "Storing Your Macros and Automatically Loading Them at Bootup," which appeared in the September 1991 issue of *Inside DOS*.

Whether you place the command in either your AUTOEXEC.BAT file or MACROS.BAT file, the command syntax is as follows:

```
doskey ff=echo ^L $g prn
```

Note that you must substitute *$g* for the greater than symbol normally used for redirection. And, as we mentioned, you'll need to press [Ctrl]P and then enter [Alt]12 on the numeric keypad to create the form feed character in the DOS Editor.

Send us your favorite tips

As a DOS user, you've probably started taking your favorite shortcut for granted. But some of your fellow *Inside DOS* subscribers might find that your tip can save them time, too. Why not share your hint with the rest of us by sending it to:

The Editor, *Inside DOS*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

If we accept your tip for publication as a letter or as the basis for an article, we'll pay you for your effort.

BATCH FILE TIP

Redirecting the output of a batch file

In previous issues of *Inside DOS*, we've shown you how to redirect a command's output to a device. For example, the DIR command normally lists onscreen the files in a directory. But you can send the output of the DIR command to the printer by adding > *prn* to the command. So, if you wanted a printed copy of your root directory, you could enter the command

```
C:\>dir > prn
```

Tacking > *prn* to the end of a command will print the output of DOS' internal and external commands, but the technique won't work with batch files. Fortunately, Contributing Editor David Reid has a workaround that lets you redirect the output of a batch file.

The trick is to run the batch file through a new, temporary copy of COMMAND.COM, DOS' command interpreter. In effect, you'll send all the output of the COMMAND command to another device. You do so by beginning your batch file command with *command /c*. Then, you type the name of the batch file. Finally, you add the redirection symbol and the name of the device to which you'll send the output.

For example, suppose you created the batch file DISKREPT.BAT, which appeared in "Determining the Amount of Disk Space Each of Your Directories Consumes" in the May 1992 issue of *Inside DOS*. This batch file displays onscreen a space usage report for directories on your hard disk. Now, let's suppose that you want to print this information instead of displaying it onscreen. You can print the space usage report by issuing the following command:

```
C:\>command /c diskrept > prn
```

Of course, you don't have to send the output to the printer—you could instead specify a filename as the destination. For example, you could issue the following command to send the space usage report to a file in the C:\LOG directory named DISKLOG.TXT:

```
C:\>command /c diskrept >> c:\log\disklog.txt
```

In this command, we used the >> redirection symbol to tell DOS to *append* the information to the file named C:\LOG\DISKLOG.TXT. If you prefer, you can use the > symbol to have DOS overwrite an existing C:\LOG\DISKLOG.TXT file.

FORMATTING TIP

DOS can't unformat backup diskettes once you reformat them

DOS 5's FORMAT command includes a safety feature that allows you to unformat the diskette. When you issue the FORMAT command, you usually see the following message:

```
Checking existing disk format.
Saving UNFORMAT information.
```

After DOS saves the unformat information, it proceeds with formatting the diskette. A MIRROR image file preserves the unformat information. This file saves the location and names of files and directories stored on the diskette before you reformatted it.

Until you store new files on the diskette, you can use the UNFORMAT command to reconstruct the old files based on the information in the MIRROR file.

DOS can unformat diskettes by using the information saved in the MIRROR file because the FORMAT command doesn't actually destroy files. Instead, the FORMAT command simply reconstructs the diskette's file allocation table (FAT), boot sector, and root directory. The FAT is a kind of index to the files on the diskette. In effect, the reformatting wipes clean the diskette's index. Of course, the MIRROR file that saves the unformat information takes up some room—as shown in Table A.

Table A

Diskette size	Space needed
360 Kb (5.25")	7,168 bytes
1.2 Mb (5.25")	12,800 bytes
720 Kb (3.5")	8,192 bytes
1.44 Mb (3.5")	13,824 bytes

This table shows how many bytes the FORMAT command needs in order to save information for unformatting a diskette.

Because DOS tries to create the MIRROR file before proceeding with the FORMAT command, a diskette must have enough room left for the MIRROR file for the process to work smoothly. If there isn't enough room on the diskette, DOS will pause and present the error message

```
Drive A error. Insufficient space for MIRROR image file.
There was an error creating the format recovery file.
This disk cannot be unformatted.
Proceed with Format (Y/N)?
```

You'll often see this message when you're reformatting backup diskettes, since the BACKUP command uses all available space. So long as you're sure that you don't need the information on the diskette, you can press Y to continue. DOS will format the diskette, but you won't be able to unformat it. If you think you might need any information on the diskette, press N. DOS will quit the FORMAT command without asking whether you want to format another diskette.

LETTERS

RAM disks can speed copying files

I enjoyed the two articles concerning RAM disks in the December 1992 issue of *Inside DOS*. (See "RAM Disks: Not Quite the Best of Both Worlds" and "Placing Temporary Files on a RAM Disk Can Speed Up Some Applications.") However, I'd like to suggest another use for RAM disks: copying a diskette. The DISKCOPY command will allow you to make an entire copy of a diskette in the same drive. But, if you use DISKCOPY, you'll have to remove the source diskette and insert the target diskette several times.

My technique is to copy all my files from the source diskette to the RAM drive, then copy the files back to the destination diskette. For example, suppose you want to copy a 3.5" diskette in the B: drive to another 3.5" diskette. Let's also suppose you've set up a RAM disk as your D: drive. You can copy all the files from the source diskette to the formatted destination diskette by following these steps:

1. Put the source diskette in the B: drive.
2. Enter the command *copy b:*.* d:*.
3. Remove the source diskette and put the destination diskette in the B: drive.
4. Enter the command *copy d:*.* b:*.

Melvin Billik
Midland, Michigan

Thanks for sharing your technique with us. Mr. Billik's method can indeed save you from the hassle of swapping diskettes with the DISKCOPY command. In some cases, copying files to a RAM disk, then to another diskette can actually give you much better copies than DISKCOPY can. Since a RAM disk can transfer data much faster than hard disks, making and copying the intermediate copies is relatively quick. Rather than copy files, DISKCOPY copies each sector of the source diskette. Consequently, if files are fragmented on the source diskette, they'll also be fragmented on the copied diskette. In fact, DISKCOPY will even copy bad sectors from the source diskette.

Mr. Billik's method is great for copying files from a diskette's root directory. If your source diskette also contains subdirectories, simply use the XCOPY command with the /E switch instead of the COPY command. (The /E switch tells XCOPY to copy all subdirectories from the diskette—even empty ones.) For example, you can use the following command to copy files from a diskette in the B: drive to a RAM disk designated as the D: drive:

```
C:\>xcopy b:\*.* d: /s /e
```

Simply reverse the *b:* and *d:* to copy files to a formatted diskette:

```
C:\>xcopy d:\*.* b: /s /e
```

If your RAM disk already contains files or subdirectories, you can set up a separate directory to hold the files from the diskette. For example, you could create a D:\DISKETTE directory on the RAM disk by entering the command

```
C:\>md d:\diskette
```

Then, be sure to specify D:\DISKETTE as the target directory in the XCOPY commands, as shown below:

```
C:\>xcopy b:\*.* d:\diskette /s /e
C:\>xcopy d:\diskette\*.* b: /s /e
```

In most situations, the XCOPY command with the /E switch copies a diskette as well as the DISKCOPY command can, if not better. But you should always use DISKCOPY if the source diskette contains hidden or system files. If you aren't sure whether a diskette contains

such files, issue the following command:

```
C:\>dir b:\ /a:h /a:s /s
```

Here, the /A:H switch tells DOS to list files with the Hidden attribute, and the /A:S switch tells DOS to list files with the System attribute. The /S switch tells DOS to look through all files in all subdirectories for files matching the hidden and system files.

Although you could theoretically use the ATTRIB command to turn off the files' hidden or system attribute, you'll save time by simply using the DISKCOPY command. In fact, DISKCOPY is the *only* choice for duplicating boot diskettes.

You'll also need to use DISKCOPY for backing up software installation diskettes. Even if they don't contain hidden or system files, most installation programs check the diskette's volume label to make sure you've inserted the proper diskette. The COPY and XCOPY commands won't transfer the diskette label; only DISKCOPY will.

One more note of caution: If you plan to place other files on your RAM disk, you should delete the diskette's files from the RAM disk after you've copied them. This additional step helps ensure you don't run out of space on the RAM disk. (If you've set the TEMP environment variable to the RAM disk, you might even encounter application errors when the disk is full.)

MS-DOS 5 won't let you boot from the B: drive

Why can't I boot from a system diskette inserted into my B: drive? DOS 5 allowed me to create a system diskette in my 3.5" B: drive by using the *format /s* command, but DOS won't boot from it. I created an identical disk in my 5.25" A: drive, and it boots the system properly. I've tried this procedure on my 386DX at home and on a Compaq 286 at work. Neither machine will boot from the B: drive.

Mike Worthington
Mukilteo, Washington

Mr. Worthington brings up an excellent point: Although MS-DOS will allow you to make a system (or *boot*) diskette in any drive, there's no guarantee you'll be able to boot from the drive. According to Microsoft Technical Support, Microsoft's version of MS-DOS 5.0 supports booting from the A: or C: drives only. However, computer manufacturers that license MS-DOS from Microsoft may make changes that will allow the operating system to boot from another drive, such as the B: diskette drive. For example, Zenith has changed the MS-DOS 5.0 software that it installs on its computers to allow you to boot from the B: drive.

If you've upgraded to DOS 5, the question is moot. Since you're running Microsoft's "pure" version of the operating system, you can boot only from the A: or C: drive. Even with an OEM version of MS-DOS that allows you to boot from a drive other than A: or C:, you may have to change your system's configuration settings to make a different diskette drive bootable. See your instruction manual for more information.

No matter what configuration or DOS version you use, it's a good idea to test a boot diskette right after you create it. Simply place the diskette in the drive, close the drive latch (if you're using a 5.25" diskette), and press [Ctrl][Alt][Delete]. If your PC boots from the diskette drive, you can stash the diskette away for use in an emergency.

Accessing configuration settings isn't so difficult

In the article "Preparing for Battery Failure Will Help You Get Back to Work Quickly," which appeared in the February 1993 issue, you explained how to record a computer's CMOS configuration settings. The technique is to press a key combination, such as [Ctrl][Alt][Enter], as your PC boots up.

While pressing the key combination will indeed work as your computer boots, I've found that I can access the CMOS configuration on my Dell 486 at any time by pressing [Ctrl][Alt][Enter]. Perhaps your readers might find that their computers, too, give them this flexibility in accessing the CMOS configuration settings.

James Tyson
Louisville, Kentucky

We apologize for making the section under the heading "Pressing keys during boot up" more difficult than necessary. As Mr. Tyson notes, you can access a Dell's configuration settings at any time—not just as your computer boots. In fact, most IBM-compatible computers manufactured in recent years allow you to access CMOS settings at any time, although the keystrokes required will vary. (The February article discusses these variations in more detail.) We're sorry for any inconvenience this oversight may have caused.